An Extensible Service Development Toolkit to Support Earth Science Grids

JASON COPE,[a] HENRY M. TUFO,[ab] AND MATTHEW WOITASZEK[a]

[a]Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA

[b]Computational and Information Systems Laboratory, National Center for Atmospheric Research, Boulder, CO 80305 USA

ABSTRACT

Grid-enabled Earth Science applications and tools are beginning to use web services to integrate distributed resources and legacy applications. Unfortunately, each application requires substantial effort to implement Grid functionality before addressing application-specific requirements. In an effort to help Earth Science application developers more rapidly develop these web services, we have created an extensible service provider toolkit (ESP). The toolkit provides the foundation to develop specialized services for Earth Science Grids, including legacy application and computational resource services. To demonstrate the functionality of ESP, we re-developed several existing web services and illustrate ESP's benefits of reduced software development time and software reuse.

1 Introduction

Grid solutions have been proposed to make managing complex workflows easier for Earth Scientists to manage and execute. However, these solutions are often:

- developed separately and in different software environments
- difficult to integrate with other Grids
- difficult to extend for other uses.

To eliminate the repetition of Grid-enabling legacy applications and accelerate service development, we have created an extensible service provider toolkit (ESP). This toolkit, consisting of Java base classes and WSDL definitions, provides extensible application control, data transfer, and application execution functionality using the Globus Toolkit[2]. By using ESP,

- the legacy Grid application developer may focus on application-specific logic instead of re-implementing simple Grid tasks
- we can achieve a 1.2 - 6x reduction in developed software over our previous service development methods
- the performance overhead of the software features and techniques we use are negligible when compared to other web service overhead costs.

2 Extensible Service Provider Toolkit

In order to quickly develop services that can reuse software components and leverage software relationships, we developed an extensible toolkit and library of software components called ESP that support enabling legacy applications and resources on the Grid. Our work is similar to the web component architecture[6, 7] since both areas of research focus on the development of extensible, specialized, and reusable components and services.

ESP is composed of four libraries of components that can be extended for a specific applications needs:

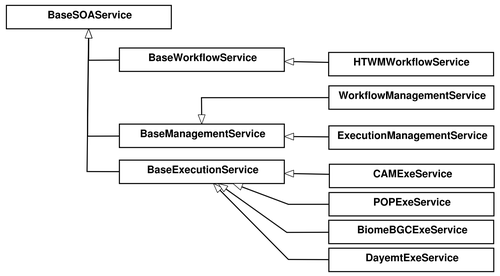
- a collection of web services that manage various Grid tasks
- a collection of web service resources that store dynamic data related to the tasks
- a communication library that provides tools for describing interactions with the services
- an interface library that provides methods for invoking the services.

These libraries contain the WSDL and Java source to implement the various services or provide additional support.

2 Extensible Service Provider Toolkit (cont.)

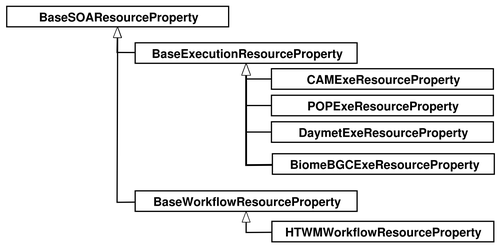
2.1 Web Service Library

We observed several operational relationships during development of several legacy application services for our SOA. We leveraged these relationships to develop extensible application management, workflow management, and service management service hierarchies (see the following UML diagram).



2.2 Web Service Resource Library

Similar to the web service library, we leveraged structural relationships in the web service resource properties to create an extensible library of web service resource properties (see the following UML diagram).



2.3 Communication Library

The goal of the communication library is to create a flexible and expressive communication framework for use by Grid applications and services. We use object marshaling techniques to communicate data between services. This library provides

- a hierarchy of base messages for developers to extend more complex messages from
- methods to marshall or unmarshall message objects to and from strings
- a generic message class that dynamically re-constructs or generates messages based on an XSD definition of the message class and simplifies the definition of operations in the service WSDL.

2.4 Interface Library

We developed an interface library to support dynamic, maintainable, and scalable client-service integration. The generic interface included in this library can invoke operations of services that fit the current ESP service model. The supporting classes provide the query and execute operations necessary to find, bind to, and execute ESP-based web services.

3 Case Study: Applying ESP

Using ESP, we developed application services to manage common legacy Earth Science applications, including Daymet[4], Biome-BGC[5], the Parallel Ocean Program (POP)[3], and the Community Atmosphere Model (CAM)[1]. We evaluated these ESP-based services for software development and invocation overhead.

3 Case Study: Applying ESP (cont.)

3.1 Extensible Service Development

We evaluated the software development overhead of ESP by analyzing the total amount of source code required to implement the legacy application services using ESP. To evaluate the WSDL definitions of the services, the total number of XML elements and attributes required to implement the application services was counted and are displayed in the following table as (element, attribute) tuples.

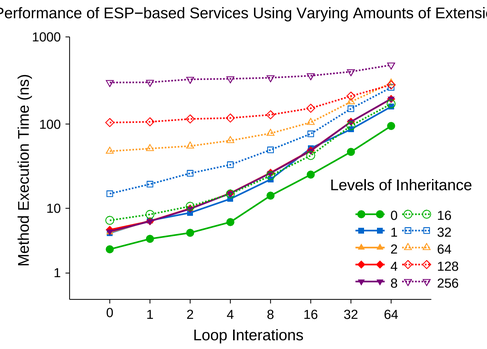
	ESP	Legacy	Improvement ($\frac{Legacy}{ESP}$)
BaseSOA	(98, 159)	-	-
BaseEXE	(90, 149)	-	-
Daymet	(27, 45)	(165, 282)	(6.11x, 6.27x)
POP	(27, 45)	(165, 282)	(6.11x, 6.27x)
BiomeBGC	(27, 45)	(165, 282)	(6.11x, 6.27x)
CAM	(27, 45)	(165, 282)	(6.11x, 6.27x)
Total	(296, 488)	(660,1128)	(2.23x, 2.31x)

To evaluate the web service implementations, the total amount of Java source lines of code (SLOC) required to implement the services was measured.

	ESP	Legacy	Improvement ($\frac{Legacy}{ESP}$)
BaseSOA	497	-	-
BaseEXE	408	-	-
Daymet	533	1095	2.05x
POP	444	915	2.06x
BiomeBGC	615	1088	1.77x
CAM	447	914	2.04x
Total	2944	3539	1.20x

3.2 Performance Analysis

The execution performance of ESP-based services was also analyzed to determine the impacts that specialization and extension have on execution performance. We developed a simple service that executed a linear computational kernel and we studied the performance characteristics of this ESP-based service using varying workloads and amounts of inheritance.



From these tests, we found that

- there is a minimal amount of overhead when small amounts of extension are used (1 - 16 levels)
- there is a noticeable amount of overhead when large amounts of extension are used (32 - 256 levels), but it is quickly amortized as the amount of work is increased
- other web service costs, such as HTTP (20ms) and HTTPS (117ms) invocation overhead, outweigh the software overhead of ESP.

4 Future Work

Future work for ESP includes

- adding support for web service libraries in other implementation languages (C, Python, and Perl)
- developing a more robust web service composition framework
- exploring web service choreography and orchestration tools
- applying ESP to other scientific disciplines beyond the Earth Sciences.

5 Conclusions

We have shown that our approach to support legacy Earth Science applications and resources in service oriented Grids through the use of an extensible service provider toolkit known as ESP

- can decrease the amount of software to develop or maintain
- does not impose a significant overhead to execute when used in moderation
- imposes execution overhead that is negligible when compared to other costs of web service invocation.

Acknowledgements

University of Colorado computer time was provided by equipment purchased under DOE SciDAC Grant #DE-FG02-04ER63870, NSF ARI Grant #CDA-9601817, NSF sponsorship of the National Center for Atmospheric Research, and a grant from the IBM Shared University Research (SUR) program. NASA has provided funding for the Grid-BGC project through the Advanced Information Systems Technology Office (NASA AIST Grant #NAG2-1646) and the Terrestrial Ecology Program.

References

[1] Collins, W., Rasch, P., Boville, B., Hack, J., McCaa, J., Williamson, D., Kiehl, J., Briegleb, B., Bitz, C., Lin, S., Zhang, M., and Dai, Y. Description of the NCAR Community Atmosphere Model (CAM 3.0). NCAR Tech. Note NCAR/TN-464+STR, 226 pp. 2004.

[2] Globus. The Globus Project, 2006, www.globus.org.

[3] Parallel Ocean Program User Guide, v 2.0. Los Alamos National Laboratory, 2003.

[4] Thornton, P.E., S.W. Running. An improved algorithm for estimating incident daily solar radiation from measurements of temperature, humidity, and precipitation. Agricultural and Forest Meteorology, 93: 211-228, 1999.

[5] Thornton, P.E., Law, B.E., Gholz, H., Clark, K., Falge, E., Ellsworth, D., Goldstein, A., Monson, R., Hollinger, D., Falk, M., Chen, J., and Sparks, J. Modeling and measuring the effects of disturbance history and climate on carbon and water budgets in evergreen needleleaf forests. Agriculture and Forest Meteorology 113 (2002), 185-222.

[6] Yang, J. and Papazoglou, M., Web Component: A Substrate for Web Service Reuse and Composition, Proceedings of the 14th Conference on Advanced Information Systems Engineering, 2002.

[7] Yang, J. Developing a framework for analyzing service composition, reuse and specialization. Communications of the ACM. Vol 46, 10. pp 35 - 40. 2003.